PROCEEDINGS

55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

Crossing Borders within the ABC

Automation,

Biomedical Engineering and

Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

Impressum
Published by

Publisher:	Rector of the Ilmenau University of Technology Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff
Editor:	Marketing Department (Phone: +49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de) Faculty of Computer Science and Automation (Phone: +49 3677 69-2860) Univ.-Prof. Dr.-Ing. habil. Jens Haueisen
Editorial Deadline:	20. August 2010
Implementation:	Ilmenau University of Technology Felix Böckelmann Philipp Schmidt

USB-Flash-Version.

Publishing House:	Verlag ISLE, Betriebsstätte des ISLE e.V. Werner-von-Siemens-Str. 16 98693 Ilmenau
Production:	CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts
Order trough:	Marketing Department (+49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de)

ISBN: 978-3-938843-53-6 (USB-Flash Version)

Online-Version:

Publisher:	Universitätsbibliothek Ilmenau ilmedia Postfach 10 05 65 98684 Ilmenau

Home / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

MDD TOOL EVALUATION FOR EMBEDDED SYSTEMS WITHIN THE AUTOMATION DOMAIN

Jirka Scheufler

University of Applied Sciences Jena
Carl - Zeiss - Promenade 2, 07745 Jena

ABSTRACT

The steadily increasing and hardly controllable complexity of modern software systems, as well as economic requirements like decreasing development-cycles within software-development in general and the automation domain in particular raise the necessity of new engineering methods [1]. The model driven development (MDD) approach promises to effectively handle these demands using platform-independent models of higher abstraction levels for software development, reusable transformation scripts and (automated) code generation out of these models [2]. Within this paper the automation domain requirements onto MDD tools and their integration into a tool chain are presented. The features of modern MDD tools are correlated with the requirements of the automation domain. An interoperability matrix is presented, which enables the selection of UML-tools to be integrated into a MDD tool chain. Within a concrete case study a tool chain was set up, generating code starting with behavioral models.

Index Terms— oAW, openArchitectureWare, MDA, OMG, UML, metamodel, meta-metamodel

1. INTRODUCTION

MDD is well established within industrial software development and addresses the use of models for software development on an abstraction level above code level, the model to model transformation within different development steps and the model to code transformation for application implementation using reusable transformation scripts. Within the automation domain MDD usage is manifold and has to be tailored to the needs of different development units[3]. This is based on the special requirements of the automation domain. This domain aims towards automating processes, e.g. those of chemical plants, paper mils and oil rigs, and ranges from very small 8-bit microcontrollers equipped with just a few kilobytes of RAM up to systems comparable to PC-platforms. While software in the PC domain is developed for a given platform, the embedded development process in the automation domain includes hardware (HW) and software (SW)[3]. Furthermore embedded system development in the automation domain needs to deal with the following issues.

- Corporate Culture leading to the distribution of development process to various regions and countries, each with its own social and technological background [4].
- Safety. Embedded devices are integral to the production process. For example, a single sensor can cause a steel-plant to shutdown. To avoid the high costs of such an event special reliability requirements are needed, such as IEC 61508 [4].
- Due to the high uptime of embedded systems, 10 up to 20 years are a common, strict standardization, and documentation, is needed to ensure HW and SW interchange, integration, upgrade, and support.
- Restrictions in scale, cost and power usage necessitate efficient code and therefore an efficient programming language without a resource hogging overhead. C, and C++ fulfill this requirement perfectly and are commonly used in embedded development[5],[6].

These issues result in a strong tool dependency for each development step, as well as the need for tool interchangeability within the toolchain to avoid lock-in-effects. This forms the base for the tool evaluation made in section 3. In section 2 the state-of-the-art of MDD is explained. Based on the evaluation results in section 4 a brief case study is presented in section 5 and furthermore discussed in the last section.

2. STATE OF THE ART

The main goal of MDD is to provide components for software development. The long-term objective, and therefore state of the art, is an end-to-end tool chain that allows the building and verification of models, as well as the generation of various artifacts or executable code from them [2]. Preferably this should happen in a homogeneous environment. Because of standardization demands, support of Object Management Group's

This work was part of a diploma thesis written at the ABB Corporate Research Center in Ladenburg.

(OMG) Unified Modeling Language (UML) based on their Meta Object Facility (MOF) [7] specifications is not required but recommended. Open source approaches like the Eclipse Modeling Framework (EMF), which is based on the Ecore metamodel and respectively the Essential Meta-Object Facility meta-metamodel (EMOF) [8], as well as Eclipse UML2 [9], based on EMF, are also possible. The challenges presented by these goals are the availability of efficient and flexible model editors which have the capability to verify the model while building them, as well as transforming and modifying them. Another challenge is the generation of code and its integration with non-generated code, as well as the support of all MDD paradigms. Based on a requirements elicitation phase for the automation domain the following capabilities for development tools in the MDD development have been analyzed [5], [10]:

- Model browsing and navigation: Analog to modern IDE's, the possibility of interactive model browsing, zooming, grouping, searching, and following references is mandatory to MDD tools.
- Consistency checking: As a system can be modeled from different points of view, for instance as state machine and interaction model, a (horizontal) consistency check is needed.
- Anti-patterns: Help to prevent the use of nice looking but bad solutions for a problem.
- Model validation: Test the static and the behavioral portions of a model against the requirements.
- Conventional (code) debugging with UML: Debug the model instead of the code and use the benefits of graphical design by masking the source code and concentrating on the problem itself.
- Metrics: Measure the quality of a model corresponding to certain criteria like complexity, cohesion, stability, testability and so on.
- Understanding model changes: Make model differences explicit and/or visualize them. This is also a requirement for the evolution of systems through different development stages.

The toolchain should also be flexible and adaptable, as well as support MDD paradigms like [2]:

- Reusability: Single models or packages should be separately exportable and importable.
- Interoperability: The models, packages and artifacts should be usable in different tools from different vendors.
- Plattform Independence: As technical progress advances the platform changes. It shall be possible to generate code for a new platform out of an "old" model without the need to change it.

3. EVALUATION OF MDD TOOLS

Based on the special requirements within the automation domain and the state of the art mentioned in sections 1 and 2, nine MDD tools were selected. A preselection was made by using different MDD-tool-lists available in the Internet (e.g. [11], [12], [7]) and the following criteria:

- As UML (standardised in ISO/IEC 19501) evolved to a *de facto* standard in model driven software development, UML must be supported. In consideration of the changes made in UML 2.X, support of UML 2.X is required.
- Due to the special requirements in the automation domain mentioned in section 1 C/C++ support in code generation or an adaptable generator engine is mandatory.
- Support of component-models, class-models and statechart-models is required for MDD.
- XMI import and export function as a basis for interoperability.

Following the tool selection a criteria matrix was developed and benchmarked during evaluation. Each of the evaluated attributes was weighted based on the overall importance. The weights were analyzed and defined in relation to the requirements for the automation domain. The following list represents the evaluation criteria with the weights of each criterion.

- **Interoperability (5):** As requested in MDD, a good interoperability is mandatory.
- **MOF-conform Metamodel (4):** Due to OMG's UML as an international standardized modeling language, the metamodel should be MOF-conform.
- **Supported Diagram Types (4):** In embedded development class-diagrams, state-diagrams and component-diagrams are mandatory to describe the embedded system.
- **UML-Notations (3):** To alter the metamodel and adapt it according to specific requirements, it should support stereotypes, tagged values and constraints.
- **Team Support (4):** To support corporate culture the tool should have support for distributed development.
- **Help Support (4):** A professional help support improves productivity and is mandatory.
- **Usability: (3):** Complex menus, hidden features and complicated handling can slow down the development process.

Requirement	Valuation	Weighting	Result
Interoperability	4	5	20
MOF-conform	4	4	16
Diagram Types	4	4	16
UML-Notations	4	3	12
Team Support	4	4	16
Help Support	3	4	12
Usability	3	3	9
Integration	4	1	4
V/E Management	1	1	1
M2M transformation	0	5	0
Extensibility	2	4	8
Artifact Generation	3	4	12
Testability	3	4	12
Overall Result:			138

Fig. 1. IBM Rational Rhapsody Evaluation Result

XMI Export Program \ XMI Import Program	Enterprise Architect	StarUML	Poseidon	Artisan	EclipseUML	MagicDraws	PowerDesigner	Objecteering	Rational Dev.
Enterprise Architect	✓	o	x	?	x	✓[a]	o	x	✓[b]
StarUML	x	x	x	?	x	x	x	x	x
Poseidon	o	o	✓	?	x	x	x	x	x
Artisan	x	x	x	?	x	x	x	x	o
EclipseUML	x	x	x	?	x	x	x	x	x
MagicDraws	x	x	x	?	x	✓	o	x	x
PowerDesigner	o[c]	x	x	?	x	o	x	x	o
Objecteering	x	x	x	?	x	x	x	x	x
Rational Developement	x	x	x	?	x	o	x	x	✓

Fig. 2. Interoperability Matrix

- **Integration: (1):** For a homogeneous environment, integration in other environments such as Eclipse or MS Visual Studio, as well as integration of e.g. Ant or Make would be suitable.
- **Version/Error Management (1):** Corresponding technolgies/tools should be transparently included in the tool.
- **Model Transformation (5):** The included generator should support M2M transformation to realize platform independence.
- **Extensibility (4):** As the MDD approach is different from the Computer Aided Software Engineering (CASE) approach, and due to flexibility needs, the generator should be extensible.
- **Artifact Generation (4):** The generator should be able to generate code (M2Text) with protected regions for usercode, as well as artifacts usable for testing purposes.
- **Testability (4):** The model shall be testable, as well as the code. Preferably, the generator should also be testable.

The evaluation was made using the three diagram types also used in the example. These diagram types represent the most important ones for embedded software development: class-diagrams and state-diagrams for system description, as well as component diagrams for HW and SW interconnection. These diagrams have been modeled in each tool and afterwards exported using the XML, as well as imported in each of the other tools. As an example figure 1 shows the evaluation result for IBM Rational Rhapsody.

4. EVALUATION RESULTS

The main result of the evaluation is the missing interoperability and interchangeability. While OMG provides with XML metadata interchange (XMI) an standardized interface for data (respectively model) interchange [7], different interpretations of OMG's UML standard, as well as proprietary extensions or constraints prohibit interoperability and interchangeability. Another problem area is the code generation engine. The tools support codegeneration out of class models. All of the tested tools claim to support MDD. Altering the codegeneration engine in some cases was impossible, which is in fact not a MDD approach but a computer aided software engineering (CASE) approach. Three of the evaluated tools (StarUML, Poseidon, EclipseUML) do not offer the needed team engineering support.

5. EXAMPLE MDD APPLICATION

As a result of the evaluation and the specific requirements of a given project, e.g. a toolchain using openArchitectureWare (oAW) [13] as a transformer engine, Enterprise Architect was choosen for the case study. For other requirements or case studies within the embedded domain, other tools would eventually be better choices. Since oAW only supports an Ecore compatible model, the three diagram types created for the evaluation within Enterprise Architect were first transformed from a MOF compatible model to an Ecore model. This was done by using EA_Xmi2Exporter [14] which is freely available via the Internet. The main challenge with the exporter is that, at this moment, not all diagram types respectively not all features are supported. Using the Xtend and Xpand component from oAW, as well as a separately developed template file for code transformation, the C++ sourcecode was generated. The sourcecode was implemented on the target (MSP430) using Zylin Embedded CDT[15]. The entire transformation and implementation process took place within the Eclipse environment.

6. SUMMARY AND LESSONS LEARNED

During the evaluation and the case study it turned out that MDD in the automation domain is feasible, but there are still some open challenges within the MDD approach. First of all a strict implementation of the OMG standard is a prerequisite. Without this, the risk of the unwanted vendor-lock-in would have to be mitigated. A solution to this would be a proprietary toolchain using the Eclipse environment, an Eclipse UML/EMF modeling tool and the oAW code generating engine with the drawback of high development and maintenance efforts. The case study revealed that oAW is a powerful and capable tool which perfectly addresses reusability, model validation and platform independence requirements (section 2). The drawback of oAW is its ”one-way-transformation”. Once the code is generated, there is currently no possibility to map the code to the model. Without such traces, debugging on a model level will not be possible.

7. REFERENCES

[1] Volker Gruhn, Daniel Pieper, and Carsten Röttgers, “MDA (R),” Springer-Verlag, 2006.

[2] Thomas Stahl, Markus Völter, and Sven Efftinge, “Modellgetriebene Softwareentwicklung. Techniken, Engineering, Management,” d.punkt Verlag, 2007.

[3] Detlef Streitferdt, Georg Wendt, Philipp Nenninger, Alexander Nyßen, and Horst Lichter, “Model Driven Development Challenges in the Automation Domain,” in *COMPSAC*. 2008, pp. 1372–1375, IEEE Computer Society.

[4] Detlef Streitferdt and Philipp Nenninger, “Quality Assurance Challenges in the Industrial Automation Domain,” in *Business Process Engineering, Proceedings of the CONQUEST 2007, d.punkt Verlag*, 2007, pp. 175–182.

[5] Jirka Scheufler, “Model Driven Development of Embedded Systems,” Diploma thesis, University of Applied Sciences Jena, Faculty of Computer Engineering, Feb 2010.

[6] Jörgen Karlsson, “Evaluation Report: Using C++ in Embedded Systems,” Tech. Rep., ABB SECRL CEWE Control, 2007.

[7] Object Management Group, “http://www.omg.org,” Last Visited: 2010.05.23.

[8] Eclipse Modeling Framework Project, “http://www.exlipse.org/modeling/emf,” Last Visited: 2010.05.23.

[9] Eclipse UML2, “http://www.eclipse.org/uml2/,” Last Visited: 2010.05.23.

[10] Yael Shaham-Gafny and Shiri Kremer-Davidson, “MDD Enablement Tooling,” White paper, IBM Research, Haifa, RAD Technolories, 2004.

[11] OOSE Innovative Informatik GmbH, “http://www.oose.de,” Last Visited: 2010.05.23.

[12] Mario Jeckle, “http://www.jeckle.de/umltools.html,” Last Visited: 2010.05.23.

[13] openArchitectureWare, “http://www.openarchitectureware.org,” Last Visited: 2010.05.23.

[14] EA_Xmi2Exporter Homepage, “http://uml2ea.blogspot.com/2007/06/connecting-server-based-ea-model-is.html,” Last Visited: 2010.05.23.

[15] Zylin Embedded CDT, “http://opensourcezylin.com/embeddedcdt.html,” Last Visited: 2010.05.23.